OPEN DOMAIN QUESTION ANSWERING

Caroline Amalie Fuglsang-Damgaard (s164175)[1], *Amalie Due Jensen (s160503)*[1],
Ida Riis Jensen (s161777)[1], *Elisabeth Zinck (s164204)*[1]

[1] Technical University of Denmark, Department of Applied Mathematics and Computer Science

ABSTRACT

When solving Open Domain Question Answering (Q&A), a good retriever that selects the correct top-k documents is of paramount importance for finding the correct answer. Traditionally, sparse vector space models have been used for retrieval, but with the invention of the transformer it is now computationally feasible to use dense representations. In this work the pre-trained *Bidirectional Encoder Representations from Transformers* (BERT) is fine-tuned to retrieve documents from a Wikipedia knowledge base. The retriever uses a simple single encoder framework to make dense representations, which are used to calculate the similarities between documents and questions in order to retrieve the most relevant documents. For 1024 question and answer pairs the Dense Passage Retriever achieves an accuracy of 91.11% when retrieving 20 documents, which outperforms a TF-IDF retriever (61.12%). This project shows the large potential of fine-tuning pre-trained models based on transformers to solve Open Domain Question Answering.

1. INTRODUCTION

The aim of Open Domain Question Answering (Open Domain Q&A) is to answer factual questions using a knowledge base consisting of a large collection of documents, e.g. Wikipedia. Thus, a model solving an open domain Q&A task is a model that can answer any question using said knowledge base. This can lead to many useful applications, such as chatbots or AI assistants. A common strategy to solve this task is to use a two-stage framework: A retriever that retrieves the most relevant documents given the question, and a reader that reads the selected documents and finds the answer.

This project focuses on the first stage of the two-part process: The retriever. Traditionally, sparse vector space models such as Term Frequency-Inverse Document Frequency (TF-IDF) have been used for retrieval, but they have now been outperformed by dense representations. An advantage of using dense representation compared to TF-IDF is that sentences with the same meaning but written with completely different words may possibly be mapped to vectors closer to one another than when using TF-IDF. The paper *Dense Passage Retrieval for Open-Domain Question Answering* by Karpukhin et. al [1] presents Dense Passage Retrieval (DPR) which uses the power of Transformers and deep neural networks to make a dense representation of the documents from the knowledge base. More specifically it utilizes the pre-trained *Bidirectional Encoder Representations from Transformers* (BERT) for creating dense representations and the paper concluded that it is possible for dense retrieval to outperform the traditional sparse retrieval component.

On the basis of [1] this project aims to re-implement DPR and investigate its performance compared to traditional methods. Further we aim to investigate the role of the BERT model in DPR by considering its pre-trained version as presented in [2] as well as the effect of different fine-tunings of the model specifically for the task of retrieving the correct documents as part of the open domain Q&A system.

The following GitHub repository contains the code used in this project: `https://github.com/elisabethzinck/deep_learning_project´`

2. BACKGROUND

Previously, recurrent neural networks (RNNs) were the go-to for reading and understanding texts. An RNN performs its computations in a cyclic manner, and the idea is that the network should be able to use the previous computations. However, a drawback of RNNs is that they face difficulties learning long-range dependencies within the sequences. Furthermore, the sequential nature makes parallelization impossible, which makes the networks computationally slow. A slow, sequential model is not suitable for Open Domain Q&A systems which typically depend on a large knowledge base containing a huge amount of data that needs to be processed. Fortunately, a new network architecture called the transformer was introduced in 2017 in the paper *Attention Is All You Need* by Vaswani et. al [3], and this network architecture solves some of the major disadvantages of RNNs by using attention instead of recurrence. A major difference between the transformer and a traditional RNN is that the input can be passed in parallel, so instead of sequentially passing one word at a time to the model, many words are passed in parallel. A transformer consists of an encoder and a decoder, but only the encoder is relevant for this project since it is the key component in the BERT model crucial to the DPR. The BERT model, as used in this project, is presented in detail in section 3.3, but first it is important to touch upon the encoder component of the transformer model.

2.1. The Encoder Component of the Transformer

Before an input is passed to the encoder, each word is mapped to a so-called *embedding space* using learned embeddings. The idea of the embedding is to capture some of the meaning of the inputs by placing semantically similar inputs close to each other in the embedding space. For further explanation of embedding space, see [3]. Moreover, a positional encoding is calculated. This is necessary because the position of a word in a sentence impacts the meaning of the word, and this positional information needs to be input to the transformer model which is non-sequential in nature. Hereafter, the input is ready for being passed into the encoder block which consists of two different layers: First a multi-head attention layer and then a feed forward layer.

The first layer of the encoder is the multi-head attention block that computes an attention vector for each word of the input. An attention vector can capture different semantics of a sentence such as: How relevant is each word to other words in the sentence? What noun does this adjective refer to? And generally; What part of the input should focus be on? Each computed attention vector will capture different semantics and if only one attention vector is computed for each word, then the attention may very likely weight its relation to the word itself as most important. However, this is not really useful information. In order to also capture the interaction between the different words of the input sentence, several intermediate attention vectors are computed for each word of the input. Combining these several attention vectors into one element has lead to the name *multi-head* attention. The multi-head attention then ultimately yields one final attention vector for each word computed as the weighted average of intermediate attention vectors.

The second layer in the encoder is a feed forward layer. In this layer, a feed forward neural network (FFNN) is applied to each of the attention vectors. The attention vectors are independent, and therefore they can be passed to the FFNN in parallel. This layer is used in order to make the output ready for being passed into a new encoder or decoder.

When combining several of such encoder components, we arrive at the so-called BERT model (see Section 3.3).

3. METHODS

In order to give an overview of the process we are presenting the methods used in the project in different subsections. First we will describe the pre-process of data along with the setup of the DPR architecture. These sections are followed by a further dive into the BERT model and lastly the models used as baseline for performance evaluation are presented.

3.1. Wikipedia Data Pre-processing

For this project we used the data set TriviaQA [4] downloaded from Huggingface's GitHub [5]. The data set contains a collection of 95K Q&A pairs with the features *question*, *question_id*, *question_source*, *entity_pages*, *search_results* and *answer* where we have used the features *question*, *question_id*, *wiki_context* from *entity_pages* and then *answer*. The documents used as the knowledge base is the collection of Wikipedia texts (in the feature *wiki_context*). Each of the Wikipedia documents is split into 128-word paragraphs where punctuation is stripped and words are set to lowercase. Given a question, q_i, the task is to find the *positive paragraph*, p_i^+, among the Wikipedia documents for question i i.e. the paragraph that can answer the question [1]. The TriviaQA data set does not contain information about where in the Wikipedia documents the answer can be found, and hence this passage of the text, i.e. the positive paragraph, needs to be defined as part of the pre-processing of the data. In order to do so, we need to measure the similarity between questions and paragraphs. Defining a target containing both question and answer, we use the function `TfidfVectorizer` from `sklearn` [6] to compute the similarity between the target and each of the paragraphs. The paragraph with the highest similarity that contains the answer is selected as the positive paragraph for the given question.

3.2. Dense Passage Retriever Architecture

The retrieval component in the Open Domain Q&A system is re-implemented as a DPR. The aim of DPR is for each document to index all paragraphs in a low-dimensional space in order to retrieve the top k paragraphs that are relevant to the input question [1]. The division of paragraphs is explained in 3.1. DPR uses a BERT model to generate dense representations of the questions and paragraphs. The BERT model acts as the dense encoder for both questions and paragraphs and is denoted $E(\cdot)$. For DPR, to retrieve the passage containing the correct answer to each question, the similarity between the paragraphs (p) and the questions (q) is computed using the dot product between the dense vectors:

$$\text{sim}(q, p) = E(q)^T E(p). \qquad (1)$$

The resulting similarities are used for determining the top k paragraphs for each question based on the highest similarities (i.e. the smallest distances).

3.3. BERT Models

To generate the dense representations of the text we used a pre-trained BERT model, which is made available to us via

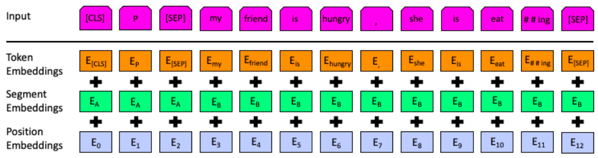

Input
[CLS] P [SEP] my friend is hungry , she is eat # # ing [SEP]
Token Embeddings
E[CLS] EP E[SEP] Emy Efriend Eis Ehungry E, Eshe Eis Eeat E# # ing E[SEP]
Segment Embeddings
EA EA EA EB EB EB EB EB EB EB EB EB EB
Position Embeddings
E0 E1 E2 E3 E4 E5 E6 E7 E8 E9 E10 E11 E12

Fig. 1: The 3-fold input scheme of the BERT model. The Classifier token [CLS] is appended in front of each input sentence, and the [P] token is added by us to illustrate that this is a paragraph not a question.

Huggingface and compatible with the `PyTorch` framework. [7]. We used the uncased BERT$_{\text{base}}$, which is a network of 12 BERT layers, i.e. 12 encoder components from the transformer architecture. Each of the encoder components have 768 hidden units and as they are encoder components of a transformer they consist of two sub-layers; An attention layer, and a dense layer. Both of these sub-layers have layer normalization and a dropout rate of 10%.

The pre-trained BERT$_{\text{base}}$ model has a vocabulary of approximately 30000 words. The input of a BERT model is the sum of a 3-fold input with word, segment and position embeddings. The special input scheme is illustrated in figure 1 and the original input scheme is thoroughly described in [2]. Especially relevant for our investigation is the classifier (CLS) token, which is appended in front of each sentence. This token makes it possible to get a dense representation of the input sentence by using the latent representation given as the hidden state corresponding to the token. Thus in order to use BERT for DPR we extract the last hidden state of the CLS token of either a passage or a question passed through the model.

For computational reasons, we use a single BERT model to encode both questions and paragraphs, which is in contrast to the approach in [1], where two separate encoders are trained (one for questions and one for paragraphs). In an attempt to make the encoder behave differently for questions and paragraphs, we have added a token, P or Q, in the beginning of the sentence indicating whether the input sentence is a paragraph or question. For questions the token is followed directly by the special separator (SEP) token indicating the segment partition of the input, while for paragraphs the token is followed by first the title of the passage's Wikipedia document and then the SEP token. To see if this encoding has any effect on the performance of our DPR system we also perform an ablation study investigating the performance of our DPR system if we do not introduce the extra tokens, but just train one BERT model to be the encoder for passages and questions alike.

3.4. Fine-tuning of BERT Models

BERT is pretrained to do Masked Language Modelling and Next Sentence Prediction which enables BERT to have a language understanding. However, when using the model for DPR we need to fine-tune BERT according to the specific task such that BERT's understanding of language can be leveraged to yield high similarities between a question and the paragraph containing the answer to the question.

We use two different methods for fine-tuning the model: 1) Fine-tuning the weights of the outermost layer (layer 11) and use the last hidden state state of the CLS token as our dense passage representation. 2) Fine-tuning the three outermost layers (layer 9-11) and concatenating the hidden states of the CLS token into one dense representation of the passage. Denoting $s(x)$ as the dense representation of a passage x, then these two approaches yield the dense representations $s(x) \in \mathbb{R}^{768}$ and $s(x) \in \mathbb{R}^{2304}$ respectively. Both of these scenarios are also carried out for the ablation study.

In order to fine-tune the BERT model for DPR we use mini-batch training and the same loss function as in [1] i.e. the negative log likelihood of a positive passage p_i^+. A positive passage within a batch is defined as the passage p_i which contains the answer to the question q_i. In our setup each question q_i in a mini batch have one positive passage p_i^+ and $n-1$ negative passages $p_{i,j}^-$ with n being the batch size. The negative log likelihood of each question is calculated as

$$L(q_i, p_i^+, p_{i,1}^-, \cdots, p_{i,n}^-) = -\log \frac{e^{\text{sim}\left(q_i, p_i^+\right)}}{e^{\text{sim}\left(q_i, p_i^+\right)} + \sum_{j=1}^{n} e^{\text{sim}\left(q_i, p_{i,j}^-\right)}},$$

where $\text{sim}(q_i, \cdot)$ is the similarity given in equation (1) between question q_i and a positive or a negative passage from the batch, i.e. p_i^+ or $p_{i,j}^-$.

BERT$_{\text{base}}$ have 110M parameters and even just fine-tuning the three outermost layers requires training of more than 21M parameters. Limited available computation power means that the two types of models are trained on 1024 questions and paragraph pairs using mini batches of 16 pairs. The training is done using 4 epochs and we have used Google Colab to access a Tesla T4 GPU. We used the AdamW optimizer and a learning rate of 5e-5 as suggested in [2].

3.5. Baseline Models

Three baseline models are implemented for comparison. The first baseline model is simply to use the pre-trained BERT$_{\text{base}}$ "off the shelf" from Huggingface without fine-tuning anything. As the BERT model is pre-trained for solving next sentence prediction and masked language modeling it is not trained for the purpose of DPR. Thus, the model is used as a baseline model such that we can investigate whether the fine-tuning of BERT increases the accuracy of our passage retrieval. The second baseline model is a TF-IDF retriever implemented using the function `TfidfVectorizer` from `sklearn` Python library. The TF-IDF retriever transforms the questions and corresponding paragraphs to two document-term matrices and from these the similarity based on the TF-IDF representation is computed using the dot product. Additionally, a random retriever is implemented in order to investigate whether our DPR performs better than retrieving k paragraphs at random.

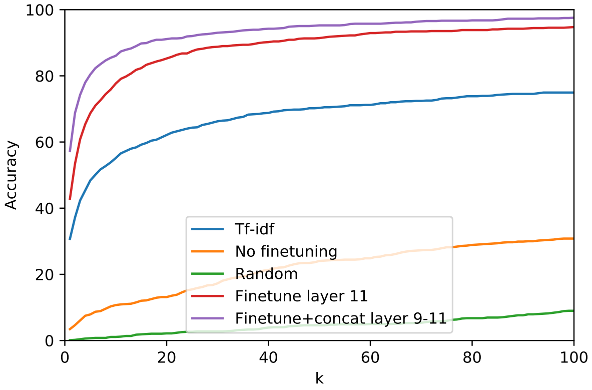


Fig. 2: Accuracy of the retrievers as a function of the number of paragraphs returned.

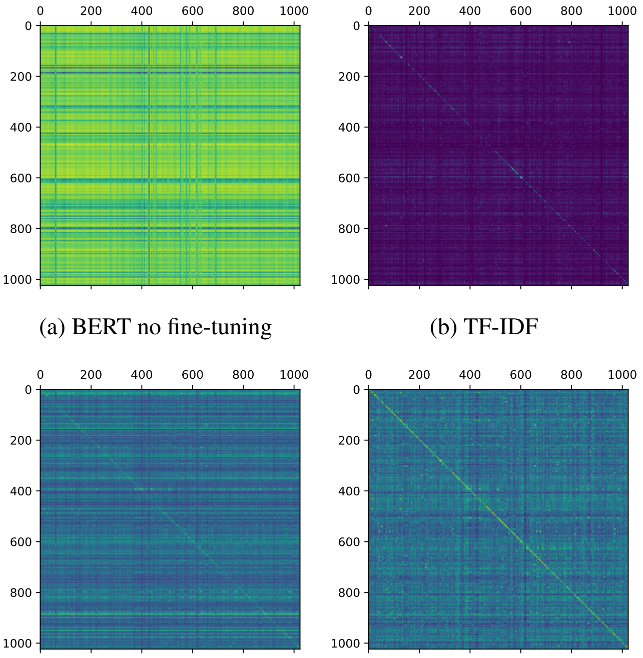
4. RESULTS

The five different retrievers; random retrieval, TF-IDF retrieval, BERT with no fine-tuning, BERT with fine-tuned 11^{th}layer, and BERT with fine-tuned and concatenated 9-11^{th}layer; were all tested on 1024 Q&A pairs from the validation set of TriviaQA. Figure 2 shows the accuracy of the retrievers when retrieving the top-k paragraphs with k ranging from 1 to 100. The simple TF-IDF retriever outperforms BERT with no fine-tuning by a large margin; for $k = 20$ BERT and TF-IDF achieve an accuracy of 13.13% and 62.12% respectively. When fine-tuning the last layer (layer 11), the accuracy increases dramatically to 85.25% for 20 paragraphs, and fine-tuning + concatenating from the 9-11^{th}layer increases the accuracy even further to 91.11% for 20 paragraphs. Table 1 shows the accuracies for different versions of the DPR for $k = 5, 20$ and 100. The best model, where layer 9 to 11 are fine-tuned and concatenated, achieves an impressive accuracy of 91.11% when allowed to retrieve $k = 20$ out of the 1024 paragraphs. The impressive improvement achieved by fine tuning the BERT model is further highlighted in figure 3. This figure illustrates the similarity matrices between the 1024 Q&A pairs using the four non-random retrievers. In the matrices each row is a question, each column is a paragraph, and in the diagonal the similarity between a question and the corresponding positive paragraph is seen. Lighter colors indicate higher similarities, while darker blue colors are lower similarities. It is seen how the better performing models have more visible light diagonals.

The two models in the ablation study, where tokens ’Q’ and ’P’ are not added before the paragraphs, perform much worse than when adding the two tokens only achieving accuracies of 60.20% and 41.72% for fine-tuning layer 11 and 9-11 respectively.

Fine-tuning	**Top-5**	**Top-20**	**Top-100**
None	7.88	13.13	30.81
Layer 11	68.69	85.25	94.75
Layer 9-11	80.40	91.11	97.58
Layer 11 (ablation)	43.33	60.20	78.79
Layer 9-11 (ablation)	25.05	41.72	63.33

Table 1: The accuracy of the models, when the top 5, top 20 and top 100 paragraphs are returned by the retriever.



(a) BERT no fine-tuning

(b) TF-IDF

(c) BERT fine-tuning 11th Layer

(d) BERT fine-tuning 9-11th layer

Fig. 3: Similarity matrices between the 1024 test Q&A pairs for four of the implemented retrievers. In the diagonal the similarity between a question and the corresponding positive paragraph is seen. Lighter colors indicate higher similarities, while darker blue colors are lower similarities

5. DISCUSSION

Our results show that using a dense representation of the text outperforms sparse approaches like TF-IDF after the pre-trained BERT model has been fine-tuned.

The poor performance of the DPR without any fine-tuning emphasizes the paramount importance of good fine-tuning of the models. As presented in figure 3a no fine-tuning of BERT resulted in some paragraphs having a consistently high similarity with almost all questions giving an almost checkered similarity matrix. After fine-tuning, this problem was somewhat alleviated and the behavior of these paragraphs was more similar to that of the others yielding the similarity matrices from figure 3c and 3d that do not have the checkered pattern. Experiments also showed that masked predictions became worse after fine-tuning, indicating that the fine-tuning alters the representations in the hidden states substantially.

The ablation study shows that concatenating P and Q to the paragraphs and questions respectively may allow for the BERT model to better adapt the encoding for these two types of text. Using this method could therefore potentially be a more computationally efficient method of adapting the encoder to the different types of text compared to fine-tuning two separate BERT models. A comparison of the two methods could be interesting in future work of this project.

The BERT models have been fine-tuned and evaluated on only 1024 pairs of questions and paragraphs from the training and validation set, respectively. To train and evaluate the models we used Google Colab, but more computational resources and space is needed to scale up the training and evaluation of the models. Due to the computational limitations, we could only use batches of 16 pairs, which is substantially lower than what was used in the original DPR paper [1]. The performance after training and evaluating on larger data sets is unknown which makes comparison with other results and the article [1] that inspired this project difficult.

The accuracy of the retrievers is likely to be upwards biased compared to what could be expected in a real-life example. This is because the positive passages do not represent the ground truth, but are only best guesses on the correct paragraphs. Moreover, the correct answer could be only partially contained in the paragraph, and generating the paragraphs using a sliding window is preferred. In order to get a more fair evaluation of the retrievers, it would be beneficial to compare them by comparing the accuracy of a reader reading the retrieved passages from the different retrievers.

6. CONCLUSION

Our results show that using a fine-tuned pre-trained BERT model to make a dense representation of questions and paragraphs performs better than using traditional methods. For 1024 question and answer pairs the dense passage retriever using a BERT model after fine-tuning 9-11$^{\text{th}}$layer achieves an accuracy of 91.11% when retrieving 20 paragraphs. This model outperforms the traditional TF-IDF retriever which only reaches an accuracy of 61.12%. This project shows the large potential of publicly available pre-trained transformer models such as BERT, that with only limited computational resources available can be fine-tuned to perform better than traditional retrievers and thus create a good foundation to solve the Open domain Q&A task.

7. REFERENCES

[1] V. Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih, "Dense Passage Retrieval for Open-Domain Question Answering," *arXiv e-prints*, 2020.

[2] J. Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," *CoRR*, vol. abs/1810.04805, 2018.

[3] A. Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin, "Attention is all you need," *Advances in Neural Information Processing Systems*, vol. 2017-December, 2017.

[4] M. Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer, "triviaqa: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension," *arXiv e-prints*, 2017.

[5] The Hugging Face Team, "Datasets," `https://github.com/huggingface/datasets`, 2020, Online; accessed 18 December 2020.

[6] Scikit Learn, "TfidfVectorizer," `https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html`, 2020, Online; accessed 18 December 2020.

[7] The Hugging Face Team, "BERT," `https://huggingface.co/transformers/model_doc/bert.html`, 2020, Online; accessed 18 December 2020.